View update for textual data: A bi-directional solution

Ravi Chugh
Advisor: Benjamin Pierce

View Update Problem

- When dealing with data stores, users are often only interested in a particular subset, or view, of the data.
- After the user makes modifications to a particular view, the view update problem is determining how to accurately update the data store to reflect these modifications.

Why Text Data?

- Although large data sets are usually stored in databases or XML, flat data text formats are often used for structured data, i.e., CSV
- Text data is often used for small- to medium-sized data sets, reduced file size, ease of use, and ease of transmission.
- For situations where data is stored in text data in a well-defined structure, the view update problem applies.

Ordinary String Lenses

- A simple first model for lenses on strings is to simulate a finite state transducer.
- A two-tape finite state transducer is an automaton that reads two input tapes, writes one output tape, and transitions based on what state it is in.
- If we treat the two arguments of the PUT function (modified view and original data store) as the two input tapes and the result (updated data store) as the output tape, we can encode a string lens as a series of FST transitions.

Harmony Project

- The Harmony project at Penn, led by Benjamin Pierce, is a data synchronization framework that addresses the view update problem with a programming paradigm called bi-directional programming.
- A bi-directional program, or lens, specifies two behaviors through one function definition: 1) how to extract a view from a complete data store (called the GET function), and 2) how to update the data store given the original store and modifications made to the view (called the PUT function).

Ordinary String Lenses

- A simple first model for lenses on strings is to simulate a finite state transducer.
- A two-tape finite state transducer is an automaton that reads two input tapes, writes one output tape, and transitions based on what state it is in.
- If we treat the two arguments of the PUT function (modified view and original data store) as the two input tapes and the result (updated data store) as the output tape, we can encode a string lens as a series of FST transitions.

Scenarios In Practice

Project / Restore

- In many applications, there is data in the store that gets projected away in the GET direction and restored exactly in the PUT direction.

Reordering Data

- Because a regular finite state transducer can only make one pass over an input tape, ordinary string lenses do not handle reordered data well.
- Two unacceptable examples PUT behaviors:

Similar Documents with Different Structure

- Documents often contain the same information presented differently.
- Need fixup functions to define these relationships.